HUMMING TRANSCRIPTION

Isaac Neri Gomez-Sarmiento, Shubham Gupta, Faez Amjed Mezdari

Université Laval, MILA

ABSTRACT

We explore classical machine learning and CNN-based algorithms for humming transcription using the recently introduced HumTrans dataset. We reveal inherent problems with the ground truth provided by the dataset and offer heuristics to address them. Furthermore, we compare the transcription accuracy of our solutions with several other methods and deliver state-of-the-art (SOTA) metrics through our model architecture and innovative post-processing techniques. All our code is availble at `https://github.com/shubham-gupta-30/piano_transcription`

***Index Terms*—** humming, transcription, AMT, MIR

1. INTRODUCTION

The field of Automatic Music Transcription (AMT) has made significant progress in developing algorithms that transform acoustic music signals into music notation, positioning it as the musical analogue to Automatic Speech Recognition (ASR). In the piano-roll convention, a musical note is typically characterized by a constant pitch (related to frequency), onset time (the start), and offset time (the end).

One application of AMT is humming transcription, which involves extracting musical notes from a hummed tune. This is a crucial component for melody search engines [1] and automatic music composition [2]. Such applications enable song identification by mere humming, provide a quick starting point for creating new songs, and democratize music creation for those who may not play an instrument or have disabilities. However, achieving error-free music transcription remains a complex challenge, even for professionals.

In this project, we worked with the HUMTRANS dataset [3], a novel dataset that claims to be the largest humming dataset to date. To date, limited research has been published using this dataset, with the exception of the paper describing the HUMTRANS dataset and its corresponding GitHub repository [4], which shows metrics of various models such as VOCANO [5], Sheet Sage [6], MIR-ST500 [7], and JDC-STP [8]. Our primary objective is to explore both classical and deep learning methods for humming transcription with the aim of enhancing these metrics.

1.1. Evaluation metrics

The authors of the HumTrans dataset utilize the library mir_eval [9] to evaluate the performance of transcription methods on their dataset. The primary motivation for using this library is to standardize the implementation of metrics for music transcription. More specifically, they employ the method $precision\_recall\_f1\_overlap$, which, according to the documentation, computes the Precision, Recall, and F-measure for reference vs. estimated notes. Correctness is determined based on note onset, pitch, and, optionally, offset, which the authors do not consider.

The authors consider a strict pitch tolerance of ±1 cent (mir_eval default value is ±50 cents), or in other words one hundredth part of a semitone, and the default onset tolerance of 50 ms.

Precision, recall and F1-score are metrics that depend on the definition of true positives (TP), false negatives (FN) and false positives (FP) [10].

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

$$F1\,Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{3}$$

TP: Estimated onset is within the tolerance of the ground truth onset and is within the pitch tolerance.

FP: If N estimated onsets are within the tolerance of the ground truth, only 1 will be considered TP if it's also within the pitch tolerance and the rest N-1 estimated onsets will be considered FP. This means that all ground truth onsets can only be matched once.

FN: No estimated onsets were detected within the tolerance of the ground truth onset.

1.2. Dataset Challenges

A major issue with the HUMTRANS dataset is that the ground truth onsets and offsets are not well aligned. This can be explained because in their methodology they instructed their subjects to synchronize their humming with the rhythm of the played melody, calling this approach as "self-labeling",

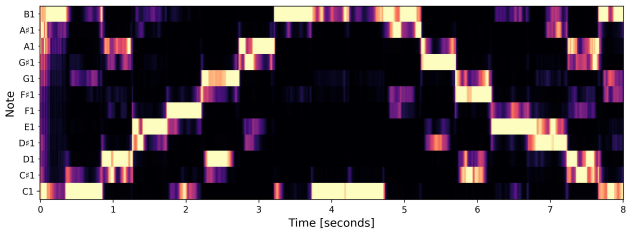

Fig. 1. Chroma features

without any post-processing. We had to overcome this challenge by coming up with semi-supervised ways to correct the provided onsets and offsets.

1.3. Octave Aware vs Octave Invariance

To simplify the problem of pitch estimation, the ground truth pitch given in MIDI file format can be transformed to an octave invariant representation by taking the modulo 12 of the MIDI numbers, which makes the song to be represented only by 12 semitones.

2. METHODS

2.1. Classical Chroma based detection

2.1.1. Chroma based features

The chroma feature is a descriptor that projects the frequency spectrum onto 12 bins, each representing how much energy of each semitone pitch class is present in the signal. Librosas's method $chroma\_cqt$ was used to get the chroma features. The frequency spectrum is obtained by the Constant-Q transform, which is useful for music signal processing, because captures best the harmonic content of the audio while being invariant to absolute frequency notes. A naive but explainable approach to pitch classification is just to take the argmax of the pitch class with the highest energy and assign a time interval the class that is most repeated.

2.1.2. Onset detection with Librosa

One of our approach to refine the ground truth onsets was to detect them using the Python library Librosa [11], specifically the method $onset\_detect$. According to the documentation, it locates onset events by picking peaks in an spectral flux onset strength envelope. The onset detection method was applied to a denoised version of the data using the library noisereduce[12, 13], which uses spectral gating. The original ground truth onsets were aligned to the estimated Librosa's onsets and matched with the nearest neighbor, to get rid of the false detections by the library. The alignment was iterative to estimate a time shift "t", which can be analogue to the Iterative Closest Point (ICP) algorithm but in 1D to estimate

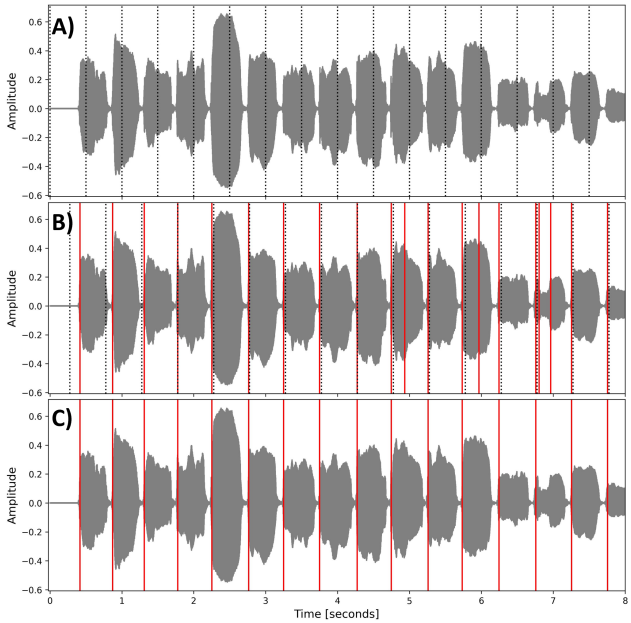

Fig. 2. Onset correction. A) Ground truth original onsets (dotted black line). B) Original onsets aligned to Librosa's estimated onsets (red line) and matched to the closest, to get rid of false detections. C) Final onset estimations.

a translation. The time shift is first initialized by computing the difference in time between the first two onsets from both sets and then iteratively adjusted by computing the mean time shift of closest onsets in both sets, until reaching a maximum number of iterations (100) or a tolerance (1e-5 s).

2.2. HMM based detection

In the Hidden Markov Model approach, The first thing we have to set is what are going to be our hidden states and our observations. The hidden states in our project are the musical notes, while the observations are derived from the spectral representation of the audio files. This spectral representation, often used in audio and speech processing, is calculated using a specific window length and step size. These parameters are determined based on the average note length in the music data. The initial probabilities, transition matrix, and observation probabilities are derived from a set of music data files. The initial probabilities represent the likelihood of a note being the first note in a piece of music, while the transition probabilities represent the likelihood of transitioning from one note to another. The number of components in the HMM is equal to the number of unique notes in the music data files modulo 12 since we are making octave invariant predications. The initial probabilities and transition matrix of the HMM are set to the ones calculated from the music data files. For the calculation of the spectral representation, we've chosen to use a single cepstral coefficient. The window length

and step size for this calculation are set to the average note length. Finally, a Gaussian HMM is trained on the spectral representations of the training data. The model is initialized with parameters that include the number of hidden states, the type of covariance, and the parameters to be estimated during training. The initial probabilities and transition matrix of the HMM are set to the ones calculated from the music data files. In the testing phase of our project, we first compute the estimated intervals. We do this by reading the audio files and calculating the duration of each file. We then read the onset times from the onset test data (We get them using the 2.5.2 method using Librosa) and create intervals between consecutive onset times. The last interval for each file is from the last onset time to the end of the audio file. Next, we use our trained HMM to predict the notes for each test file. We calculate the MFCCs for each file and use the HMM to predict the hidden states, which correspond to the notes. The predicted notes are then stored for later use. We also extract the reference pitches and intervals from the MIDI files. For each note in the MIDI file, we store the pitch and the start and end times of the note. These serve as the ground truth against which we will compare our predictions.

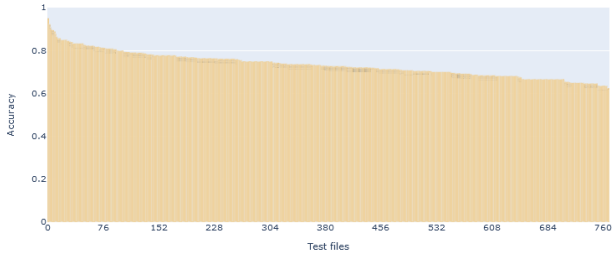

Fig. 3. Comparison of predicted notes and reference notes for each test file using the HMM

2.3. Neural Network based detection

2.3.1. Better ground truth annotation

For the purpose of training a neural network, we need precise onsets and offsets for ground truth. We realized that more accurate labeled onsets and offsets result in better-trained models.

To this end, we designed a heuristic-based algorithm to compute improved onsets and offsets. This algorithm calculates a *waveform envelope* and determines onsets and offsets based on when this envelope dips below a specific threshold value. The onsets and offsets obtained in this way can be noisy, so we refine them by eliminating spurious onsets/offsets, enforcing a minimum note length, and maintaining a minimum silence length between notes. This method is supervised by using the number of notes from the ground truth provided by the dataset. We retain only those training, testing, and validation samples where the number of notes detected by our heuristic matches the number provided by the ground truth. It's important to note that we can only trust the number of notes from the ground truth, as the onsets and offsets cannot be relied upon. By using this approach, we obtained better ground truth onsets and offsets, retaining $6,827$ of the $13,080$ in the training set (52.2%) and 440 of the 769 (i.e., 57%) in the test set.

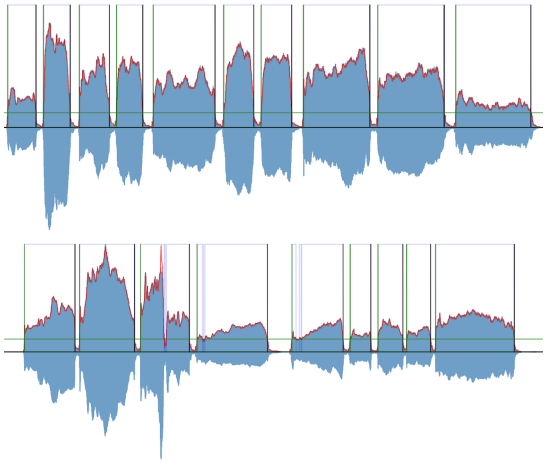

Fig. 4. Examples where the heuristic algorithm for onset and offset detection succeeds (top) and fails (bottom).

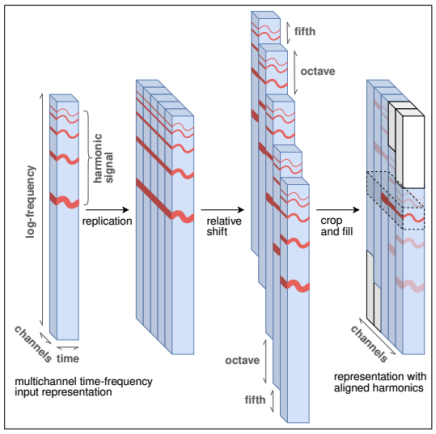

Fig. 5. Harmonic Stacking. Source: [14]

2.3.2. Network Design

Our neural network architecture is a convolution-based network. The model is inspired by the architecture in [15]. We have tailored our network for our use case of a monophonic humming dataset. The following are key elements of this design:

- **Input representation**: While $STFT$ and Mel spectrograms work well for speech-related tasks, CQT

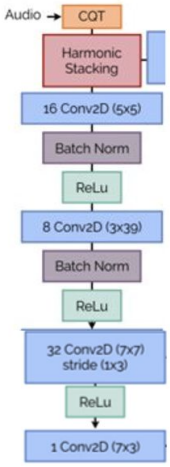

Fig. 6. Model Architecture

representation is much more effective for MIR (Music Information Retrieval) as the geometric structure of this transform closely matches the geometric nature of Western classical music.

- **Harmonic Stacking**: While instruments can produce desired notes precisely, humans aren't very adept at doing so. Human singing/humming normally includes not only the main note but also overtones, which can be spatially far apart from each other in a CQT transform. To address this, [14] introduces Harmonic Stacking, where CQT time frames are shifted by the right amount of offsets to bring overtones closer to each other

2.3.3. Training

We train the model with a batch size of 16. For each batch, we randomly select a small sample from the humming recordings by choosing 5 to 10 notes for each batch element. Additionally, we introduce a new dummy note 89 to signify the beginning and end of the recording sample, as well as the silence between notes. Our task for every time frame of the CQT representation is to predict the note that the frame represents. The model is trained using **CrossEntropy** loss and employs **Adam** as the optimizer with a learning rate of 0.001. Unlike other works in this field that first predict onsets and offsets and then condition note prediction on them, as in [2], our method infers onsets and offsets directly from the predicted notes.

2.3.4. Inference

Unlike transformers, convolution networks generalize well beyond the training length examples they are trained on. Because of this, we do not have to perform inference on pieces of a sample being inferred; instead, we can perform inference

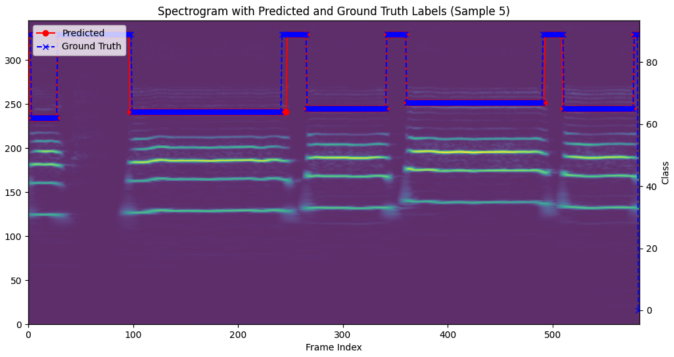

Fig. 7. Example inference - blue is the ground truth and red is infered cleaned notes

on the entire sample as long as available memory allows. During inference, we calculate model logits for each time frame over the possible space of notes. The naive way to convert these logits to actual note predictions would be to take the note at each time frame with the maximum probability. However, this results in noisy note attributions.

We clean up these noisy attributions using a dynamic programming based algorithm inspired by the use of the *Viterbi algorithm* in text-to-speech alignment in speech tasks. Given the affinity matrix denoting the affinity scores of each time frame with the possible notes, we can define a path P through this matrix as valid if it satisfies the following constraints (Note that the dummy note introduced is 89, and T is the length of the segment being inferred):

- P starts at $(0, 89)$ and ends at $(T-1, 89)$.
- If P is at note $n \neq 89$ at time t, then at time $t+1$, it can be at either n or 89.
- If P is at note 89 at time t, then it can be at any note at time $t+1$.

These constraints ensure that we do not switch abruptly from one note to another without going through the dummy note, which is a realistic constraint as in all humming samples, the space between two hummed notes is very noticeable. Using a dynamic programming-based method, we can find the path P with the highest probability among all valid paths. We further clean up this path P to enforce minimum note length constraints and report this final cleaned note assignment as our inferred note assignment. We read the onsets and offsets from these note assignments.

3. MIDI TO MUSIC SCORE

After extracting pitches and their time intervals from a MIDI file, we can generate a music score using the Music21 library. We convert MIDI numbers to note names (e.g., C4, D5) and

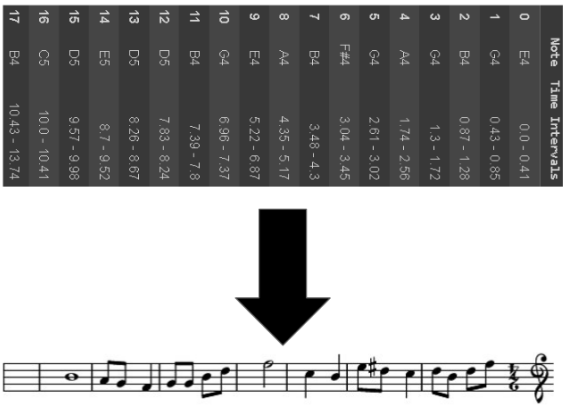

Fig. 8. An example using the file $F01\_0024\_0001\_1.mid$

write these, along with their start and end times, to a text file. Each line in this file represents a note. We then read this file, creating a Music21 note for each line and adding it to a Music21 score. To visualize this score, we convert it to a LilyPond file. LilyPond, a music engraving program, uses this file to create a high-quality image of the music score. Finally, we convert the LilyPond file to a PNG file to obtain our music score.

4. RESULTS AND DISCUSSION

We compare our methods with various methods discussed in [3]. The authors provide extracted MIDIs for the test set for 4 methods in their GitHub repository [4]. These are - $Vocano$ [5], $MIR-ST500$ [7], $SheetSage$ [6], and $JDC-STP$ [8]. In addition, we compute MIDIs for the test set using Spotify's $basic\_pitch$ [2], and compare these with the methods proposed in this report.

4.1. Octave invariant

Following [3], we calculate precision, recall, and F1-scores, using the mir_eval library, with an onset tolerance of $50ms$ and disregarding offsets. We provide two comparisons here - a comparison with respect to the corrected ground truth we obtain and a comparison where we measure only the note accuracy, disregarding both onsets and offsets. Additionally, these comparisons are octave invariant, ie.e a note is considered to be correctly predicted even if the ocatve does not match the ground truth exactly. We provide these results on the test set provided by the dataset in table 4.1

4.2. Octave aware

In table 4.2, we also provide comparisons of our CNN-based method with other methods while requiring the models to be octave-aware, i.e., we are looking for an exact note match, including the correct octaves.

Method	Note + Onsets			Notes Only		
	P	R	F1	P	R	F1
Chroma feature	0.452	0.630	0.519	0.580	0.85	0.68
HMM	0.230	0.209	0.218	0.670	0.617	0.640
CNN	**0.670**	**0.675**	**0.673**	**0.848**	**0.854**	**0.850**
VOCANO	0.568	0.561	0.564	0.729	0.723	0.726
JDC-STP	0.502	0.487	0.490	0.795	0.784	0.783
SheetSage	0.171	0.170	0.170	0.446	0.442	0.444
MIR-ST500	0.601	0.608	0.604	0.808	0.820	0.813
basic_pitch	0.392	0.497	0.434	0.653	0.847	0.729

Table 1. Octave Invariant

Method	Note + Onsets			Notes Only		
	P	R	F1	P	R	F1
CNN	**0.649**	**0.653**	**0.651**	**0.814**	**0.820**	**0.817**
VOCANO	0.344	0.340	0.341	0.446	0.443	0.444
JDC-STP	0.297	0.279	0.286	0.463	0.442	0.450
SheetSage	0.161	0.160	0.161	0.434	0.430	0.444
MIR-ST500	0.360	0.363	0.361	0.486	0.491	0.488
basic_pitch	0.243	0.304	0.268	0.388	0.498	0.432

Table 2. Octave Aware

4.3. Discussion

We observe that our CNN based methods outperform all tracked methods for humming transcription. Also note that the chroma feature based algorithm outperforms most tracked methods when it comes to note accuracy in the octave invariant setting. Our HMM based method also cperforms fairly well when measuring note accuracy only. We observe that SheetSage performs the worst in all comparisons.

5. CONCLUSION AND FUTURE WORK

We have provided a range of solutions for humming transcription using a recently released dataset. We demonstrate that the ground truth provided by the dataset has flaws, and we utilize semi-supervised methods to establish a reasonable ground truth. Our exclusively designed CNN-based method for this problem outperforms many existing solutions for similar problems found in the current literature. We also show that methods based on chroma features and Hidden Markov Models (HMM) are very effective and come close to the performance of many deep learning-based methods according to the evaluation metrics we tracked.

While we have explored some CNN-based architectures, we believe that it is possible to develop much smaller networks without sacrificing performance for this task. This belief is supported by some of our experiments not covered in this report and by the success of the classical machine learning methods explored here. Furthermore, we are confident that our work can be expanded to the more complex task of polyphonic Singing Voice Transcription (SVT), and we plan to pursue this extension in our future work.

6. REFERENCES

[1] Google, "Song stuck in your head? just hum to search," Accessed: Dec. 12, 2023. [Online]. Available: https://blog.google/products/search/hum-to-search/.

[2] Rachel M. Bittner, Juan José Bosch, David Rubinstein, Gabriel Meseguer-Brocal, and Sebastian Ewert, "A lightweight instrument-agnostic model for polyphonic note transcription and multipitch estimation," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Singapore, 2022.

[3] Shansong Liu, Xu Li, Dian Li, and Ying Shan, "Humtrans: A novel open-source dataset for humming melody transcription and beyond," *arXiv preprint arXiv:2309.09623*, 2023.

[4] shansongliu, "Humtrans," Accessed: Dec. 12, 2023. [Online]. Available: https://github.com/shansongliu/HumTrans.

[5] Jui-Yang Hsu and Li Su, "VOCANO: A note transcription framework for singing voice in polyphonic music," in *Proc. International Society of Music Information Retrieval Conference (ISMIR)*, 2021.

[6] Chris Donahue, John Thickstun, and Percy Liang, "Melody transcription via generative pre-training," in *ISMIR*, 2022.

[7] Jun-You Wang and Jyh-Shing Roger Jang, "On the preparation and validation of a large-scale dataset of singing transcription," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 276–280.

[8] Keunhyoung Luke Kim Taehyoung Kim Sangeun Kum, Jongpil Lee and Juhan Nam, "Pseudo-label transfer from frame-level to note-level in a teacher-student framework for singing transcription from polyphonic music," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022.

[9] Colin Raffel, Brian McFee, Eric J Humphrey, Justin Salamon, Oriol Nieto, Dawen Liang, Daniel PW Ellis, and C Colin Raffel, "Mir_eval: A transparent implementation of common mir metrics.," in *ISMIR*, 2014, vol. 10, p. 2014.

[10] Sebastian Böck, Florian Krebs, and Markus Schedl, "Evaluating the online capabilities of onset detection methods.," in *ISMIR*, 2012, pp. 49–54.

[11] Brian McFee, Vincent Lostanlen, Alexandros Metsai, Matt McVicar, Stefan Balke, Carl Thomé, Colin Raffel, Frank Zalkow, Ayoub Malek, K Lee, et al., "librosa/librosa: 0.10.1," *Version 0.10.1, Zenodo, doi*, 2023.

[12] Tim Sainburg, "timsainb/noisereduce: v1.0," June 2019.

[13] Tim Sainburg, Marvin Thielk, and Timothy Q Gentner, "Finding, visualizing, and quantifying latent structure across diverse animal vocal repertoires," *PLoS computational biology*, vol. 16, no. 10, pp. e1008228, 2020.

[14] Jirí Balhar and Jan Hajic jr, "Melody extraction using a harmonic harmonic convolutional neural network," *dimensions*, vol. 10, no. 16x10x540, pp. 16x10x360.

[15] Rachel M Bittner, Juan José Bosch, David Rubinstein, Gabriel Meseguer-Brocal, and Sebastian Ewert, "A lightweight instrument-agnostic model for polyphonic note transcription and multipitch estimation," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 781–785.